#ifndef DATA\_H

#define DATA\_H

#include <pthread.h>

#include "darknet.h"

#include "darknet.h"

#include "matrix.h"

#include "list.h"

#include "image.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

#include "tree.h"

static inline float distance\_from\_edge(int x, int max)

{

int dx = (max/2) - x;

if (dx < 0) dx = -dx;

dx = (max/2) + 1 - dx;

dx \*= 2;

float dist = (float)dx/max;

if (dist > 1) dist = 1;

return dist;

}

//typedef struct{

// int w, h;

// matrix X;

// matrix y;

// int shallow;

// int \*num\_boxes;

// box \*\*boxes;

//} data;

//typedef enum {

// CLASSIFICATION\_DATA, DETECTION\_DATA, CAPTCHA\_DATA, REGION\_DATA, IMAGE\_DATA, LETTERBOX\_DATA, COMPARE\_DATA, WRITING\_DATA, SWAG\_DATA, TAG\_DATA, OLD\_CLASSIFICATION\_DATA, STUDY\_DATA, DET\_DATA, SUPER\_DATA

//} data\_type;

/\*

typedef struct load\_args{

int threads;

char \*\*paths;

char \*path;

int n;

int m;

char \*\*labels;

int h;

int w;

int c; // color depth

int out\_w;

int out\_h;

int nh;

int nw;

int num\_boxes;

int min, max, size;

int classes;

int background;

int scale;

int small\_object;

float jitter;

int flip;

float angle;

float aspect;

float saturation;

float exposure;

float hue;

data \*d;

image \*im;

image \*resized;

data\_type type;

tree \*hierarchy;

} load\_args;

typedef struct{

int id;

float x,y,w,h;

float left, right, top, bottom;

} box\_label;

void free\_data(data d);

pthread\_t load\_data(load\_args args);

pthread\_t load\_data\_in\_thread(load\_args args);

\*/

void print\_letters(float \*pred, int n);

data load\_data\_captcha(char \*\*paths, int n, int m, int k, int w, int h);

data load\_data\_captcha\_encode(char \*\*paths, int n, int m, int w, int h);

data load\_data\_old(char \*\*paths, int n, int m, char \*\*labels, int k, int w, int h);

data load\_data\_detection(int n, char \*\*paths, int m, int w, int h, int c, int boxes, int classes, int use\_flip, int gaussian\_noise, int use\_blur, int use\_mixup,

float jitter, float hue, float saturation, float exposure, int mini\_batch, int track, int augment\_speed, int letter\_box, int show\_imgs);

data load\_data\_tag(char \*\*paths, int n, int m, int k, int use\_flip, int min, int max, int w, int h, float angle, float aspect, float hue, float saturation, float exposure);

matrix load\_image\_augment\_paths(char \*\*paths, int n, int use\_flip, int min, int max, int w, int h, float angle, float aspect, float hue, float saturation, float exposure, int dontuse\_opencv);

data load\_data\_super(char \*\*paths, int n, int m, int w, int h, int scale);

data load\_data\_augment(char \*\*paths, int n, int m, char \*\*labels, int k, tree \*hierarchy, int use\_flip, int min, int max, int w, int h, float angle, float aspect, float hue, float saturation, float exposure, int use\_mixup, int use\_blur, int show\_imgs, float label\_smooth\_eps, int dontuse\_opencv);

data load\_go(char \*filename);

box\_label \*read\_boxes(char \*filename, int \*n);

data load\_cifar10\_data(char \*filename);

data load\_all\_cifar10();

data load\_data\_writing(char \*\*paths, int n, int m, int w, int h, int out\_w, int out\_h);

list \*get\_paths(char \*filename);

char \*\*get\_labels(char \*filename);

char \*\*get\_labels\_custom(char \*filename, int \*size);

void get\_random\_batch(data d, int n, float \*X, float \*y);

data get\_data\_part(data d, int part, int total);

data get\_random\_data(data d, int num);

void get\_next\_batch(data d, int n, int offset, float \*X, float \*y);

data load\_categorical\_data\_csv(char \*filename, int target, int k);

void normalize\_data\_rows(data d);

void scale\_data\_rows(data d, float s);

void translate\_data\_rows(data d, float s);

void randomize\_data(data d);

data \*split\_data(data d, int part, int total);

data concat\_data(data d1, data d2);

data concat\_datas(data \*d, int n);

void fill\_truth(char \*path, char \*\*labels, int k, float \*truth);

void fill\_truth\_smooth(char \*path, char \*\*labels, int k, float \*truth, float label\_smooth\_eps);

#ifdef \_\_cplusplus

}

#endif

#endif